Schäfer, Günter:

Sabotageangriffe auf Kommunikationsinfrastrukturen: Angriffstechniken und Abwehrmaßnahmen

URN: urn:nbn:de:gbv:ilm1-2015210290

Published OpenAccess: January 2015

Original published in:

Praxis der Informationsverarbeitung und Kommunikation : PIK. - Berlin : de Gruyter (ISSN 1865-8342). - 28 (2005) 3, S. 130-139.
DOI: 10.1515/PIKO.2005.130
URL: http://dx.doi.org/10.1515/PIKO.2005.130
[Visited: 2015-01-20]

K.G. Saur Verlag, München, 2005

G. Schäfer

Sabotageangriffe auf Kommunikationsinfrastrukturen: Angriffstechniken und Abwehrmaßnahmen

Günter Schäfer studierte von 1989 bis 1994 Informatik an der Universität Karlsruhe (TH) und war von 1994 bis 1999 wissenschaftlicher Mitarbeiter am dortigen Institut für Telematik. Nach seiner Promotion (1998) arbeitete er zwischen 1999 und 2000 im Rahmen eines 18-monatigen Forschungsaufenthaltes an der Ecole Nationale Superieure des Télécommunications (ENST) in Paris und im Anschluss daran bei dem Fachgebiet Telekommunikationsnetze an der Technischen Universität Berlin. Seit April 2005 ist er Leiter des Fachgebiets Telematik/Rechnernetze der Technischen Universität Ilmenau. Seine wissenschaftlichen Interessen liegen auf den Gebieten der Sicherheit von Kommunikationsinfrastrukturen und der leistungsfähigen Gestaltung innovativer Kommunikationsdienste und -architekturen. Prof. Dr. Schäfer ist Mitglied der ACM, der GI und des IEEE.

ZUSAMMENFASSUNG

Sabotageangriffe, im Englischen auch als „Denial of Service Attacks (DoS-Attacks)“ bezeichnet, zielen darauf ab, die Verfügbarkeit bestimmter Systeme oder Dienste für berechtigte Nutzer zu reduzieren bzw. die entsprechenden Systeme und Dienste vollständig außer Betrieb zu setzen. In diesem Beitrag wird ein Überblick über in Sabotageangriffen verwendete, grundlegende Techniken und prinzipielle Abwehrmaßnahmen gegeben.

1 EINLEITUNG

Mit der zunehmenden Integration von Informations- und Kommunikationssystemen in nahezu alle Bereiche des privaten, gesellschaftlichen und geschäftlichen Lebens steigt in modernen Informationsgesellschaften die Abhängigkeit von der Verfügbarkeit und korrekten Funktion der diesen Diensten zugrunde liegenden Kommunikationsinfrastrukturen. Als immer größere Bedrohung erweisen sich in diesem Zusammenhang vorsätzliche Sabotageangriffe auf grundlegende Kommunikations- oder Systemdienste. Zwischen den Jahren 1989 und 1995 erhöhte sich beispielsweise die Zahl der dem so genannten Computer Emergency Response Team (CERT) gemeldeten Vorfälle um 50% jährlich [11]. Eine Studie des amerikanischen FBI aus dem Jahr 1999 berichtet, dass 32% der an der Studie teilnehmenden Stellen im davorgehenden Jahr DoS-Angriffe auf Ihre Systeme festgestellt haben [12].

Diese Lage verschlimmert sich zusehends: so weisen bereits Studien aus dem Jahr 2000 darauf hin, dass Angreifer zunehmend spezifische Angriffswerkzeuge einsetzen, mit denen verteilte Angriffe, die von einer Vielzahl von Systemen ausgehen, aufgesetzt und koordiniert werden können [26]. Bei der letztgenannten Angriffskategorie spricht man im Englischen Sprachgebrauch von „*Distributed Denial of Service (DDoS)*“:

Statistiken des *Computer Emergency Response Teams (CERT)* [27] zeigen eine weitere Zuspitzung der Lage aufgrund einer ständig zunehmenden Anzahl von Angriffsvorfällen. Die Anzahl der registrierten Vorfälle stieg in den letzten vier Jahren von 21.756 im Jahr 2000 auf 137.529 im Jahr 2003 (für 2004 sind noch keine Zahlen publiziert). Darüber hinaus erweist sich auch die Diversität und in der Regel schlechte Qualität von Software als erhebliches Problem. Im Jahr 2000 wurden vom CERT 1.090 Schwachstellen in Software-Systemen registriert. Dieser Wert stieg in den Jahren 2001 und 2002 auf 2.437 bzw. 4.129 an und liegt für die Jahre 2003 und 2004 mit 3.784 bzw. 3780 aufgedeckten Schwachstellen nur unwesentlich unter dem Wert von 2002, wodurch allerdings wenigstens eine Trendwende angedeutet wird.

Dass sich eine schnelle Schließung entdeckter Sicherheitslücken per Software-Patch in der Praxis oft als illusorisch erweist, zeigten im Sommer 2003 erneut die Erfahrungen mit dem sogenannten *W32/Blaster-Wurm* [4].

Diese Entwicklungen zeigen weiterhin auch eine alarmierende Entwicklung der Ausbreitungsgeschwindigkeit von Computer-Angriffen. Nach einer Studie von Weaver [28] ist es möglich, hypervirulente aktive Wurm-Software zu konstruieren, die in der Lage ist, alle zu einem Zeitpunkt mit dem Internet verbundenen und für eine spezifische Schwachstelle empfindlichen Rechnersysteme innerhalb von 15 Minuten zu infizieren. Gemäß den Schätzungen einer anderen Studie [25] können sehr kompakte Software-Würmer unter bestimmten Umständen sogar in weniger als 30 Sekunden alle für eine spezifische Schwachstelle empfindlichen Systeme infizieren.

Mit dem erfolgreichen Eindringen in ein System können in Bezug auf Sabotageangriffe hauptsächlich zwei Zielsetzungen verbunden sein: Zum einen kann das System durch Einbringen eines Software-Wurms selbst in seinen Grundfunktionen gestört werden. Das war beispielsweise bei dem genannten W32/Blaster-Wurm der Fall. Zum anderen kann jedoch eine erfolgreiche Manipulierung eines Systems auch ein vorbereitender Schritt für weitere Sabotageangriffe auf andere Systeme sein. In diesem Fall dient der manipulierte Computer als eines von vielen sogenannten *Slave-Systemen* bei der Durchführung ei-

nes verteilten Denial-of-Service Angriffs (Englisch: *Distributed Denial of Service, DDoS*).

In den folgenden Abschnitten sollen insbesondere Sabotageangriffe auf die Grundfunktionen der protokollverarbeitenden Instanzen eines Netzes (jeweils in Vermittlungssystemen, Servern und Arbeitsplatzrechnern) beschrieben und mögliche Gegenmaßnahmen erörtert werden. Aufbauend auf der im nächsten Abschnitt eingeführten generellen Charakterisierung wird im weiteren auf Angriffe eingegangen, die:

- zum einen auf die für die reine Datenübertragung erforderliche Protokollverarbeitung abzielen, und
- zum anderen grundlegende Sicherheitsfunktionen wie Authentisierung und Schlüsselverwaltung für DoS-Angriffe ausnutzen.

2 GRUNDFORMEN VON SABOTAGEANGRIFFEN IN KOMMUNIKATIONSNETZEN

In modernen Netzen können durch die Gefahr von Sabotageangriffen unterschiedliche Ziele bedroht werden:

- spezifische Dienste, die Teilnehmern angeboten werden (z.B. Server, die bestimmte Dienste anbieten, oder Server bestimmter Firmen),
- die Kommunikationsinfrastruktur selbst, insbesondere einzelne Zugangsnetzwerke, da Bandbreite im Zugangsnetzbereich immer noch eine knappe Ressource darstellt und sich hieran in Zukunft vermutlich auch wenig ändern wird, sowie
- die Endgeräte einzelner Nutzer, vor allem durch die Tatsache verstärkt, dass sich etwa mobile Endgeräte von relativ funktionsbeschränkten Mobiltelefonen hin zu flexibel programmierbaren Kleinstrechnern („Palmtop PC“) mit Standard-Betriebsystemen entwickeln, wodurch sie in zunehmendem Maße ein attraktives Ziel für Viren, trojanische Pferde etc. darstellen.

Das einzelnen Angriffsvarianten innewohnende Risikopotential fällt hierbei umso größer aus, je mehr Systeme gegenüber der jeweiligen Angriffsart verletzlich sind. In dieser Hinsicht stellt der derzeitige Trend zur Vereinheitlichung der Kommunikationsinfrastrukturen auf Grundlage des Internet Protokolls (IP) bei allen anderen sich hieraus ergebenden Vorteilen eher eine Verschärfung der Bedrohungslage dar.

Grundlegend können Sabotageangriffe in zwei Hauptkategorien eingeteilt und die folgenden Angriffstechniken unterschieden werden:

- Bei der *Ressourcenzerstörung* ist das generelle Ziel, einen bestimmten Dienst vollständig außer Funktion zu setzen. Grundlegende Angriffstechniken hierbei sind:
  - das *Eindringen in Systeme* mit anschließender (zerstörender) Manipulation der Systeme,
  - das *Ausnutzen von Implementierungsschwächen* wie Pufferüberlauf, die zum Teil bereits zum Absturz eines Systems führen oder auch ein Eindringen ermöglichen können, sowie
  - die *Abweichung von der korrekten Protokollausführung*, welche je nach Implementierung auch einen Absturz des angegriffenen Systems hervorrufen kann.
- Die *Ressourcenerschöpfung* hingegen zielt darauf ab, bestimmte Ressourcen wie Rechenkapazität, Speicher, Bandbreite etc. eines angegriffenen Systems zu binden und die Verfügbarkeit des Systems so für die reguläre Nutzung teilweise oder vollständig einzuschränken. Im Allgemeinen werden die folgenden Angriffsformen unterschieden, die jeweils unterschiedliche Ressourcen als Ansatzpunkt verwenden:
  - das *Hervorrufen aufwändiger Berechnungen* wie beispielsweise rechenintensive kryptographische Operationen (Exponentiation modulo einer Primzahl für Diffie-Hellmann-, RSA-, oder ElGamal-Operationen),
  - die *Speicherbelegung mit (nutzloser) Zustandsinformation*,
  - die vorgetäuschte *Reservierung von Ressourcen*, wie z.B. Bandbreite in einem Kommunikationsnetz, sowie
  - die *Überlastung durch hohe Verkehrslast*, wobei dem Angreifer hierbei eine hohe Gesamtbandbreite zur Verfügung stehen muss.

3 ANGRIFFE AUF NETZWERKEBENE

In Kommunikationsnetzen können die oben genannten Angriffstechniken auf die Protokollverarbeitungsfunktionen der Schichtenarchitektur, nach der heutige Kommunikationssysteme aufgebaut sind, angewendet werden. Während einige der Angriffstechniken durch eine Kombination etablierter Maßnahmen wie sorgfältiger Systemadministration, Software-Engineering, Überwachung und Angriffserkennung (*Intrusion Detection*) in Ihrem Risiko begrenzt werden können, erfordern die Angriffstechniken Protokollabweichung und Ressourcenerschöpfung Maßnahmen, die für spezifische Kommunikationsprotokolle dediziert untersucht und integriert werden müssen. Sie sollen daher im folgenden näher betrachtet werden.

3.1 Abweichung von der korrekten Protokollausführung

Angriffe durch Abweichen von der korrekten Protokollausführung, im Englischen Sprachgebrauch auch *Protocol Deviation Attacks* genannt, nutzen in der Regel Schwachstellen in der Implementierung der protokollverarbeitenden Funktionen im Betriebssystem eines End- oder Zwischensystems aus. Je nach Implementierung können eine Reihe oder sogar ein einziges gezieltes und absichtlich nicht-protokollkonformes Paket ein System zum Absturz bringen, weswegen diese Angriffe gelegentlich auch als „*Nuke Attacks*“ bezeichnet werden. Es existiert mittlerweile ein Vielzahl solcher Angriffe, die oft jedoch auf ähnlichen Grundprinzipien beruhen, so dass an dieser Stelle lediglich einige „prominente“ Vertreter in ihrer Funktionsweise erklärt werden sollen.

Ein sehr bekanntes Beispiel ist der sogenannte „*Ping of Death*“, der aus einem an das Zielsystem gerichteten, zu großen „Echo Request“-Paket des *Internet Control Message Protocol (ICMP)* besteht. Reguläre „Echo Request“-Pakete weisen eine Länge von 64 Oktetten auf, die mögliche Länge hierfür ist jedoch nicht begrenzt. Durch das Senden mehrerer IP-Fragmente kann nun ein ICMP-Paket konstruiert werden, das erst beim empfangenden System wieder reassembliert wird und bei Überschreiten der maximal zulässigen Größe von 64.507 Oktetten je nach Implementierung der Protokollfunktionen zu einem Absturz oder Neustart des Systems führen kann. Die Überschreitung der maximalen Größe für eine Protokolldateneinheit (Protocol Data Unit, PDU) ist eine generelle Angriffstechnik, die selbstverständlich nicht auf ICMP-Nachrichten beschränkt ist, sondern auch bei beliebigen anderen Protokollen angewendet werden kann.

Die Fragmentierung und Reassemblierung von IP-Paketen ermöglicht eine Reihe weiterer Angriffsmöglichkeiten. So ist es

beispielsweise ohne weiteres möglich, eine Reihe von IP-Fragmenten zu senden und dabei ein Fragment – beispielsweise das erste, letzte oder ein beliebiges anderes Fragment des Pakets – nicht zu senden. Die IP-Protokollinstanz des empfangenden Systems wartet in diesem Fall auf das ausstehende Fragment, um das ursprüngliche Paket reassemblieren zu können. Da die bereits empfangenen Fragmente Speicher belegen, kann bei gezielter Anwendung dieser Technik eine Ressourcenerschöpfung des Empfängers erreicht werden (man erkennt an dieser Stelle, dass die Grenzen zwischen den einzelnen Angriffsformen an manchen Stellen fließend sind). Als mögliche Gegenmaßnahme sollte eine *Ratenkontrolle* in Bezug auf ausstehende Fragmente bei der Empfangsinstanz eingeführt werden, d.h. bei Überschreiten einer gewissen Rate ausstehender Fragmente verwirft die Empfangsinstanz gezielt einkommende bzw. bereits gespeicherte Fragmente (siehe auch Abschnitt 3.2).

Ein weiterer Fragmentierungsangriff wird durch das Angriffswerkzeug Teardrop realisiert, das eine früher häufig anzutreffende Schwachstelle von Betriebssystemen im Umgang mit „überlappenden" IP-Fragmenten ausnutzt. Bei der Fragmentierung großer Datagramme teilt eine IP-Protokollinstanz diese in kleinere Teile und sendet sie in mehreren IP-Paketen, wobei jeweils mit dem *More Data Flag (MF)* angezeigt wird, dass noch weitere Fragmente folgen, und in dem *Fragment-Offset-Feld* die Position jedes Fragments innerhalb des Pakets mitgeteilt wird. Diese Informationen werden vom Empfänger für die korrekte Reassemblierung benötigt.

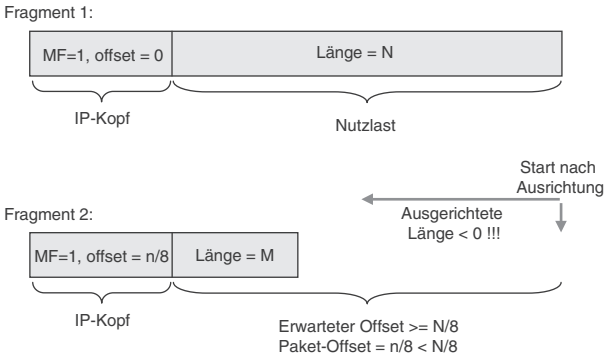


Abb. 1 Funktionsweise des Teardrop-Angriffes

Abb. 1 zeigt die Grundidee des Teardrop-Angriffs mit zwei Fragmenten. Das erste Fragment transportiert *N* Oktette und das zweite Fragment transportiert *M* Oktette. Daher wird im ersten IP-Protokollkopf das MF-Feld auf „1" und das Offset-Feld auf „0" gesetzt sein. Entsprechend ist in dem zweiten Fragment das MF-Feld auf „0" und das Offset-Feld auf den Wert *N/8* gesetzt (der Offset wird stets in Blöcken von 8 Oktetten angegeben). Der Teardrop-Angriff manipuliert das Offset-Feld des zweiten Fragments auf einen Wert *n/8* der kleiner als der korrekte Wert *N/8* ist.

Einige IP-Implementierungen versuchen, das zweite Fragment durch Anpassen des Offset-Wertes auszurichten. Ist die Länge *M* des zweiten Fragments gering, so kann es hierbei dazu kommen, dass der Beginn des zweiten Fragments hinter sein Ende gelegt wird. Bei der Ausführung des Reassemblierungsalgorithmus führt das dazu, dass eine sehr große Anzahl vermeintlicher Daten an den entsprechenden Speicherbereich angefügt wird, da die vermeintlich „negative" Länge des zweiten Fragments von dem Algorithmus als positive Zahl interpretiert wird (dieses Fehlverhalten hängt mit der Kodierung ganzer Zahlen im Zweierkomplement zusammen). Insgesamt führt das in der Regel zur Überschreibung fremder Speicherbereiche, was – da die Protokollfunktionen vom Betriebssystemkern erbracht werden – einen Systemabsturz auslöst. Es existiert eine Vielzahl weiterer Angriffswerkzeuge, die ähnliche Mechanismen verwenden, um anvisierte Rechner gezielt zum Absturz zu bringen, z.B. *Targa, SYNdrop, Boink, Nestea Bonk, Teardrop2* und *NewTear*.

Eine weitere Klasse von Angriffen ergibt sich daraus, dass eine Reihe von TCP/IP-Implementierungen dafür anfällig ist, wenn Pakete zu ihrer Quelle geleitet werden. Das kann beispielsweise dadurch erreicht werden, dass ein TCPSYN-Rahmen, der einen Verbindungsaufbauwunsch signalisiert, durch bewusste Manipulation (sogenanntes „*Address Spoofing*") gleiche Ziel- und Quell-Adressen sowie -Ports aufweist. Das Angriffswerkzeug *Land* ist ein weit verbreiteter Vertreter dieser Klasse.

Obwohl eine große Vielfalt ähnlicher Angriffe nach den beschriebenen Mustern existieren, sind die meisten von ihnen bekannt und es existieren Software-Updates zur Behebung dieser Schwachstellen für alle gängigen Betriebssysteme. Da jedoch in jedem Betriebssystem ständig neue Fehler auftreten (z.B. durch Funktionserweiterungen eingebracht werden) bzw. entdeckt werden, ist auf absehbare Zeit damit zu rechnen, dass weiterhin neue DoS-Angriffe entwickelt werden.

Als Gegenmaßnahmen gegen Angriffe dieser Art können derzeit hauptsächlich die regelmäßige Verfolgung diverser Sicherheitsnachrichten und zügige Aktualisierung der Betriebssysteminstallation bei Bekanntwerden neuer Angriffe empfohlen werden.

3.2 Ressourcenerschöpfung

Die Überlastung der Kapazitäten eines Opfers durch Fluten mit einer großen Menge in böswilliger Absicht gesendeter Nachrichten ist die grundlegendste und zugleich auch am schwersten abwehrbare Form von Sabotage durch Ressourcenerschöpfung. Das liegt vor allem daran, dass die Abwehr einer hohen Verkehrslast (sogenanntes „*Dumb Flooding*") bereits im Netz, ggf. an mehreren Stellen unterstützt werden muß, um eine Verteidigung auch gegen Angreifer zu gewährleisten, denen eine hohe Bandbreite zur Verfügung steht.

Um jedoch eine möglichst hohe „Sabotage-Ausbeute" bei dem Opfer zu erreichen, versuchen die meisten DoS-Angreifer mehr Schaden als den bloßen Konsum von Übertragungsbandbreite zu erzielen. Somit wird angestrebt, mit möglichst wenig angreiferseitigem Aufwand einen maximalen Ressourcenverbrauch beim Opfer zu erzielen. Aus diesem Grund werden bevorzugt solche Pakete gesendet, die nicht nur Übertragungkapazitäten binden, sondern auch von dem Opfer ausgewertet und entsprechend verarbeitet werden müssen. Führen solche Pakete weiterhin zur lokalen Belegung von Speicher, zu aufwändigen Berechnungen oder auch zu Antwortpaketen, so kann ein signifikanter Ressourcenkonsum ausgelöst werden.

Ein verbreitetes Beispiel für solch einen Angriff ist der sogenannte *TCP/SYN-Flood-Angriff*, bei dem der Angreifer primär auf die Überlastung des für die Bearbeitung eingehender Verbindungen zur Verfügung stehenden Speichers abzielt. Der Angreifer sendet eine große Anzahl von TCP/SYN-Paketen, mit denen jeweils ein Verbindungsaufbauwunsch signalisiert wird,

an einen offenen Port des Zielrechners. Dabei gibt er bei jedem Paket eine andere (gefälschte) Adresse als Quelladresse an (diese Technik wird im Englischen als *Address Spoofing* bezeichnet).

Für jedes eingehende Paket durchsucht der Empfänger seine bereits bestehenden Verbindungen und belegt einen Speicherbereich für eine neue Verbindung, sofern das Paket nicht zu einer bereits bestehenden Verbindung gehört. Anschließend antwortet er mit einem TCP/SYN-ACK-Paket, um die Annahme des Verbindungsaufbauwunsches zu signalisieren. Das Ergebnis nach der empfängerseitigen Verarbeitung wird auch als *TPC-Half-Open* bezeichnet, da das antwortende System nun auf die Bestätigung des erfolgreichen Verbindungsaufbaus wartet.

Falls die gefälschte Quelladresse des ursprünglichen TCP/SYN-Pakets auf ein in dem Moment mit dem Internet verbundenes System verweist, so empfängt dieses System nun ein TCP/SYN-ACK-Paket, zu dem es kein TCP/SYN-Paket gesendet hat, so dass es auch keinen entsprechenden Verbindungszustand gespeichert hat. Aus diesem Grund wird es dem angegriffenen System mit einem TCP/RST-Paket antworten, worauf dieses den vermeintlichen Verbindungszustand wieder löscht. In dem Fall, dass kein System zu der gefälschten Adresse gehört bzw. in dem Moment mit dem Internet verbunden ist, wird das angegriffene System solange warten, bis ein entsprechender Zeitgeber (Timeout) des TCP-Protokolls abläuft, so dass der für den Verbindungszustand belegte Speicher für einen längeren Zeitraum nicht anderweitig genutzt werden kann.

Tab. 1 zeigt die protokollkonformen Antworten auf diverse Angriffspakete unter der Voraussetzung, dass kein entsprechender Zustand – also beispielsweise eine offene TCP-Verbindung – bei dem empfangenden System vorhanden ist.

Tab. 1 Antwortpakete gemäß Protokollspezifikation

Gesendetes Paket	Antwort des Empfängers
TCP SYN (zu offenem Port)	TCP SYN/AC;
TCP SYN (zu geschl. Port)	TCP RST (ACK)
TCP ACK	TCP RST (ACK)
TCP DATA	TCP RST (ACK)
TCP RST	keine Antwort
TCP NULL	TCP RST (ACK)
ICMP Echo Request	ICMP Echo Reply
ICMP TS Request	ICMP TS Reply
UDP Paket (zu offenem Port)	protokollabhängig
UDP Paket (zu geschl.Port)	ICMP Port Unreachable

In Folge gefälschter Quelladressen werden weitere Rechner zu sekundären Opfern des Angriffs. Für diesen Effekt unerwarteter Pakete wurde im englischen Sprachgebrauch der Begriff *Backscatter* geprägt (vergleiche auch Abb. 2). In [18] wird eine Untersuchung beschrieben, wie auf der Grundlage dieses Effekts eine Abschätzung über die Menge und Intensität von DoS-Aktivitäten im Internet vorgenommen werden kann.

Weiterhin können die als Reaktion versendeten, vermeintlichen Antwortpakete auch gezielt für indirekte DoS-Angriffe verwendet werden: Bei dieser Variante sendet der Angreifer eine große Anzahl an Paketen an unterschiedliche Empfänger, wobei er jeweils die Adresse des eigentlichen Angriffszieles als Quelladresse angibt. Das Opfer wird dann mit der Verarbeitung von Antwortpaketen überlastet, die aus dem ganzen Internet stammen. Bei einigen Angriffen dieser Art sendet der Angreifer gezielt an Broadcast-Adressen (z.B. der Form x.y.z.255), so dass je nach Protokollimplementierung der per Broadcast angesprochenen Systeme pro Angriffspaket eine Vielzahl von Paketen an das eigentliche Opfer geschickt wird. Man nennt die vermeintlich antwortenden Systeme hierbei *Reflektoren*. Angriffe nach diesem Muster werden auch als *Reverse Attacks* bezeichnet und können als eine (einfache) Form sogenannter *verteilter Sabotageangriffe (Distributed Denial of Service, DDoS)* betrachtet werden (siehe auch weiter unten). Die bekanntesten Angriffe dieser Art sind unter den Namen *Smurf* (auf der Grundlage von ICMP-Paketen) und *Fraggle* (verwendet UDP-Pakete) bekannt.

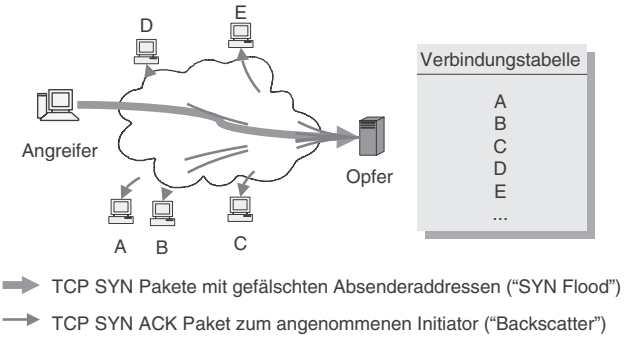


Abb. 2 Sogenannter Backscatter als Reaktion auf einen „SYNFlood“-Angriff mit gefälschten Absenderadressen

3.2.1 Distributed Denial of Service (DDoS)

Im Laufe der Zeit sind die Strategien für DoS-Angriffe zunehmend komplexer geworden. Als besonders schwer abzuwehrende Variante erweisen sich hierbei *verteilte Sabotageangriffe (DDoS)*. Hierbei nutzt der Angreifer die geballte Kapazität einer Vielzahl koordiniert agierender Systeme, wodurch selbst sehr leistungsfähige Serversysteme mit einer breitbandigen Internet-Verbindung von solchen Angriffen in ihrer Verfügbarkeit bedroht werden können. Zusätzlich zu dem eigentlichen Angreifer und Opfer sind in einen DDoS-Angriff meist noch sogenannte *Master-* und *Slave-Systeme* eingebunden (vergleiche Abb. 3). Als eigentliche Angreifer gegenüber dem Opfer agieren die Slave-Systeme, die von dem Angreifer jedoch nicht direkt sondern mittelbar über die Master-Systeme koordiniert werden.

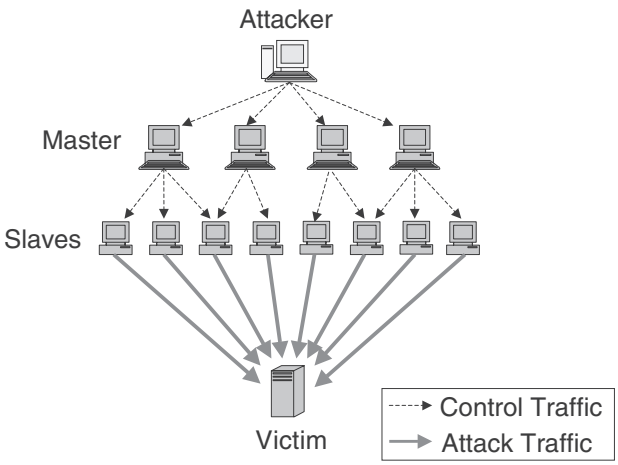
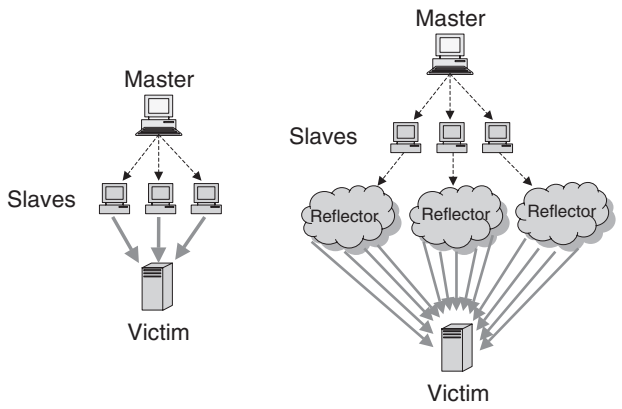


Abb. 3 Allgemeine Struktur eines DDoS-Angreifer-Netzes

Aus der Konstruktion eines DDoS-Angreifer-Netzes gemäß Abb. 3 ergeben sich für den Angreifer eine Reihe von Vorteilen: zum einen ist seine Identität schwerer zu ermitteln, da der Verkehr selbst durch direktes Rückverfolgen nicht mehr zu ihm führt, und zum anderen wird die Stabilität des DDoS-Netzes durch die zusätzlichen Master-Systeme weiter erhöht, weil selbst bei Bekanntwerden eines Master-Systems maximal die von diesem System kontrollierten Slave-Systeme aufgedeckt und außer Betrieb gesetzt werden können. Das DDoS-Netz wird somit gegen teilweisen Ausfall resistent. Um die mit einem solchen Netz zu erzielende Angriffsintensität zu erhöhen, wird ein Angreifer versuchen, als Slaves möglichst breitbandig angebundene Systeme einzusetzen.

Die für solche Angriffsnetze benötigten Systeme beschafft sich ein Angreifer üblicherweise zuvor durch Kompromittierung unzureichend geschützter und gewarteter Systeme, auf denen er im Anschluss teilweise modifizierte Systemprogramme – sogenannte *Rootkits* – installiert, um seine Spuren und die Existenz der Master- bzw. Slave-Software zu verbergen [8]. Die von den Slave-Systemen ausgeführten Angriffe sind meist übliche DoS-Angriffe, wie TCP/SYN-Flood, Smurf, Fraggle etc.



a) Master-Slave-Victim b) Master-Slave-Reflector-Victim

Abb. 4 Topologien unterschiedlicher DDoS-Varianten

Zur zusätzlichen Steigerung der Angriffsintensität können die Slave-Systeme wiederum Reflektor-Netze nutzen, so dass weiterhin die in Abb. 4 gezeigten Angriffstopologien unterschieden werden können. Bei Nutzung von Reflektor-Netzen wie im rechten Teil von Abb. 4 gezeigt senden die Slave-Systeme spezifische Pakete (z.B. ICMP-Echopakete) an die Broadcast-Adresse der Reflektor-Netze, wobei als Quelladresse jeweils die Adresse des eigentlich angegriffenen Systems eingesetzt wird. Jedes System der angesprochenen Reflektornetze antwortet darauf dem angegriffenen System mit einer protokollkonformen Nachricht. Durch die hohe insgesamt beim Opfer ankommende Verkehrsmenge kann es zu einer vollständigen Auslastung der verfügbaren Internet-Anbindung oder zumindest der Verarbeitungskapazität des Opfersystems kommen. Zu beachten ist bei dieser Angriffsvariante weiterhin, dass der bei dem Opfersystem ankommende Verkehr stets korrekte Quell- und Zieladressen aufweist, da die Systeme der Reflektor-Netze in der Annahme, dass sie zuvor von dem angegriffenen System ein Paket erhalten haben, mit ihrer tatsächlichen Quelladresse senden.

3.2.2 Gegenmaßnahmen

Für eine effektive Abwehr ressourcenerschöpfender Sabotageangiffe wurde bisher eine Reihe von Gegenmaßnahmen vorgeschlagen, die in die folgenden Kategorien eingeteilt werden können:

- *Generelle vorbeugende Maßnahmen* zielen darauf ab, potentiellen Angreifern die Realisierung von Sabotageangriffen möglichst schwierig zu gestalten. Hierzu gehören beispielsweise Paketfilterregeln zur Erkennung gefälschter Quelladressen (beispielsweise im Zugangsnetzbereich eines Internet Service Providers installiert), aber auch regelmäßige Software-Aktualisierungen, um bekannte Sicherheitslücken zu schließen und somit Angreifern das Eindringen in Systeme zu erschweren. Als weitere vielversprechende generelle Gegenmaßnahme erweist sich die Einführung einer generellen *Ratenkontrolle* für ressourcenkonsumierende Protokollabläufe (z.B. Bearbeitung eingehender Verbindungsaufbauwünsche). Hierbei werden bei Überschreiten einer festgelegten Auftragsrate (Signalisiervorgang, Verkehrsmenge etc.) überzählige Aufträge nach einem zufälligen Muster verworfen, um die generelle Arbeitsfähigkeit eines Systems auch unter stark ansteigender Last zu erhalten.
- *Angriffserkennung* ist eine offensichtliche Voraussetzung für eine effektive Verteidigung vor Sabotageangriffen. In der Praxis erweist sich diese einfach zu stellende Aufgabe jedoch oft als schwieriger als sie erscheint.
- *Gezielte Abwehrmaßnahmen* umfassen alle Maßnahmen, um die Auswirkungen eines erfolgenden Angriffs zu begrenzen. Hierbei können drei Schritte unterschieden werden:
  - Maßnahmen zur akuten Angriffsabwehr in der Nähe des Opfers wie Paketfilterung, Konfigurationsänderung etc.,
  - Maßnahmen zur Rückverfolgung des Angriffs (Englisch: *Traceback*) mit dem Ziel der Angreiferidentifikation, sowie
  - Unterbinden des Angriffs an der Quelle (im Idealfall; z.B. durch Abschalten des Netzzugangs des Angreifers).

Im Hinblick auf Gegenmaßnahmen im akuten Angriffsfall ist zu bemerken, dass sich die Abwehr eines einmal begonnenen DoS-Angriffs relativ schwierig gestaltet. Das ist insbesondere dann der Fall, wenn die Überlastung bereits durch die bloße Verkehrsmenge eintritt. Gegebenenfalls kann durch Einwirken auf die Routerkonfiguration und Paketfilter eine Linderung der Angriffsintensität erreicht werden. Aufgrund der Tatsache, dass bei DoS-Angriffen oft gefälschte Quelladressen eingesetzt werden, und der hohen Verkehrsintensität im Fall eines verteilten Sabotageangriffs ist es in der Realität oft nicht möglich, eine schnelle Linderung eines solchen Angriffs zu erzielen.

Da die meisten Sabotageangriffe mit gefälschten Quelladressen arbeiten, ist die Rückverfolgung der tatsächlichen Verkehrsquelle eine vordringliche Aufgabe, für die bereits eine Reihe von Ansätzen vorgeschlagen worden sind [22, 24].

4 ANGRIFFE AUF SICHERHEITSFUNKTIONEN

Als eine weitere Möglichkeit Sabotageangriffe einzuschränken kann die Verwendung kryptographischer Protokolle zur Identitätsprüfung vor der eigentlichen Diensterbringung angesehen werden, so dass ausschließlich die Anfragen authentisierter Dienstnehmer bearbeitet werden müssen. Aus der Tatsache, dass kryptographische Protokolle oft berechnungsintensive Verarbeitungsschritte enthalten, ergibt sich jedoch ein neues

DoS-Risiko, nämlich das von Sabotageangriffen auf Sicherheitsfunktionen eines Systems. In diesem Abschnitt werden daher DoS-Angriffe auf Authentisierung und Schlüsselaushandlung beschrieben und Ansätze zu ihrer Unterbindung erörtert.

4.1 Grundlagen zur Authentisierung

Die Authentisierung stellt den grundlegendsten Sicherheitsdienst dar, da die meisten anderen Sicherheitsdienste auf ihr aufbauen. Hierbei sind die folgenden beiden Varianten zu unterscheiden [23]:

- Mit *Nachrichtenauthentisierung (Data Origin Authentication)* wird der Sicherheitsdienst bezeichnet, der eine Überprüfung des Erzeugers einer Nachricht sowie ihrer Unversehrtheit ermöglicht. Somit kann zu einem späteren Zeitpunkt überprüft werden, ob eine Nachricht bzw. allgemeiner eine Menge von Daten noch genau den Inhalt hat, den ihr Erzeuger ursprünglich vorgegeben hat. Ein synonymer Begriff hierfür ist *Nachrichtenintegrität (Data Integrity)*.
- Die *Instanzenauthentisierung (Entity Authentication)* ist der Sicherheitsdienst, der Kommunikationspartnern die fälschungssichere Überprüfung der Identität ihrer Partnerinstanzen ermöglicht.

Im Allgemeinen kann die Instanzenauthentisierung mit unterschiedlichen Mitteln realisiert werden:

- *Wissen*, wie Passwörter etc.,
- *Besitz*, beispielsweise physikalischer Schlüssel oder Zugangskarten,
- *Unfälschbare Eigenschaften*, also etwa biometrische Eigenschaften wie Fingerabdrücke etc.,
- *Aufenthaltsort* einer Instanz, so wird beispielsweise die Authentizität von Kassierern in Bankzweigstellen selten von Kunden überprüft, sowie
- *Delegierung der Authentisierungsprüfung*, bei der die verifizierende Instanz akzeptiert, dass eine andere Instanz ihres Vertrauens bereits eine Authentizitätsprüfung durchgeführt hat.

Da in Kommunikationsnetzen eine direkte Überprüfung mit den hier angeführten Mitteln schwierig zu realisieren bzw. unsicher ist, werden hierfür in der Regel kryptographische Protokolle eingesetzt. Wie generelle Kommunikationsprotokolle auch müssen kryptographische Protokolle eine Reihe von Eigenschaften aufweisen, um als Protokoll gelten zu können:

- Jede in den Ablauf eines Protokolls eingebundene Instanz muss das Protokoll sowie alle hierfür erforderlichen Schritte und Nachrichtenformate im Voraus kennen.
- Jede in einen Protokollablauf eingebundene Instanz muss der definitionsgemäßen Ausführung des Protokolls zustimmen.
- Das Protokoll darf nicht mehrdeutig sein, d.h., alle Schritte müssen wohldefiniert sein und es darf keine Möglichkeit existieren, den Protokollablauf misszuverstehen.
- Das Protokoll muss vollständig sein, insbesondere muss für jede mögliche Situation eine in diesem Fall auszuführende Aktion spezifiziert sein.

Weiterhin müssen kryptographische Protokolle eine zusätzliche Eigenschaft erfüllen, um ihrem Anwendungszweck gerecht zu werden:

- Es darf nicht möglich sein, etwas anderes zu tun oder in Erfahrung zu bringen, als in dem Protokoll spezifiziert ist.

Während dem letztgenannten Aspekt in den letzten 20 Jahren eine besondere Aufmerksamkeit bei dem Entwurf und der Analyse kryptographischer Protokolle zuteil wurde, ist die Forderung, dass alle beteiligten Instanzen der definitionsgemäßen Ausführung des Protokolls zustimmen, wenig in Frage gestellt worden. Neuere Entwicklungen in den letzten Jahren zeigen jedoch, dass kryptographische Protokolle ein erhebliches DoS-Risiko in sich bergen, wenn bei Ihrem Entwurf diese Voraussetzung als gegeben angesehen wird. Dieser Umstand ergibt sich insbesondere aus der Tatsache, dass die meisten kryptographischen Protokolle vorsehen, dass ein Server bei der Protokollbearbeitung relativ berechnungsintensive kryptographische Algorithmen ausführt.

Die in Authentisierungs- und Schlüsselaushandlungsprotokollen verwendeten kryptographischen Algorithmen können in die folgenden Kategorien eingeteilt werden:

- *Verschlüsselungsalgorithmen*, die weiterhin in die folgenden beiden Klassen eingeteilt werden:
  - *Symmetrische kryptographische Algorithmen* verwenden einen Schlüssel, das heißt, eine Nachricht wird mit demselben Schlüssel entschlüsselt, mit dem sie auch verschlüsselt wurde, bzw. eine Signatur wird mit dem gleichen Schlüssel überprüft, mit dem sie auch erstellt wurde. Beispiele gebräuchlicher Algorithmen dieser Klasse sind *DES, 3DES, AES* und *RC4* [17].
  - *Asymmetrische kryptographische Algorithmen* verwenden zwei verschiedene Schlüssel für die Chiffrierung und Dechiffrierung. Die beiden Schlüssel hierfür können nicht unabhängig voneinander gewählt werden, sondern müssen als Paar nach einer für den kryptographischen Algorithmus spezifischen Weise konstruiert werden. In der Praxis sind insbesondere die jeweils nach Ihren Erfindern benannten Algorithmen *RSA* [21] und *ElGamal* [9] im Einsatz.
- Algorithmen zur Berechnung *kryptographischer Prüfwerte* bestehend aus den Unterkategorien:
  - *Modifikationserkennungswerte (Modification Detection Code, MDC)*, die in gewisser Weise „kryptographische Fingerabdrücke“ von Nachrichten realisieren (wichtige Vertreter: *MD5* und *SHA1*), und
  - *Nachrichtenauthentisierungswerte (Message Authentication Code, MAC)*, die zusätzlich ein zuvor ausgehandeltes „Geheimnis“, d.h. einen Sitzungsschlüssel, in die Berechnung miteinbeziehen und somit unmittelbar die Verifikation der Authentizität einer Nachricht ermöglichen (gebräuchliche Algorithmen: *CBC-MAC* unter Verwendung einer symmetrischen Blockchiffre und *HMAC* unter Verwendung von MD5 oder SHA1).

Im Hinblick auf mögliche DoS-Bedrohungen ist zu bemerken, dass asymmetrische Veschlüsselungsalgorithmen das höchste Risikopotential aufweisen, da sie im Vergleich mit symmetrischen kryptographischen Algorithmen oder Algorithmen zur Berechnung kryptographischer Prüfwerte einen um einen Faktor 100 bis 1000 höheren Berechnungsaufwand verursachen. Das gilt in ähnlicher Weise auch für den in der oben stehenden Aufzählung nicht genannten, bei der Schlüsselvereinbarung oft eingesetzten Diffie-Hellman-Algorithmus [6] zur Aushandlung eines vertraulichen Geheimnisses über einen offenen Kanal.

4.2 Sabotageangriffe auf Authentisierungsprotokolle

Sabotageangriffe auf Authentisierungsprotokolle machen sich die Tatsache zunutze, dass die Ausführung dieser Protokolle erhebliche Ressourcen eines Servers beansprucht. Konkret müssen in der Regel pro Authentisierungsvorgang berechnungsintensive Algorithmen ausgeführt und zusätzlich Zustandsdaten, wie Identitäten, Zufallszahlen, ausgehandelte Parameter etc. gespeichert werden. Ohne zusätzliche Schutzmaßnahmen (siehe unten) ist es einem Angreifer daher möglich, durch Vortäuschen einer definitionsgemäßen Protokollbearbeitung mit geringem Aufwand eine erhebliche Menge an Ressourcen eines Servers zu binden. Abb. 5 zeigt ein Beispiel für den prinzipiellen Ablauf eines Sabotageangriffs auf die Authentisierungsfunktion eines Servers.

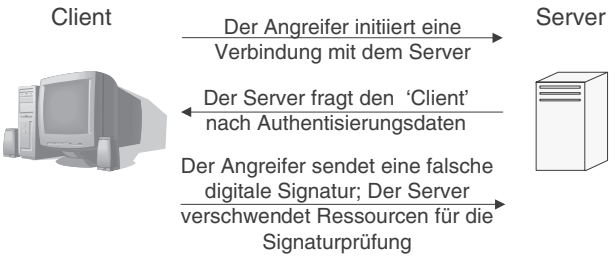


Abb. 5 Beispiel eines DoS-Angriffs auf den Authentisierungsdienst

Wie im vorigen Abschnitt erwähnt, sind insbesondere asymmetrische kryptographische Algorithmen sehr berechnungsintensiv, so dass alle kryptographischen Protokolle, die von diesen Algorithmen Gebrauch machen, potentiell von Sabotageangriffen durch Ressourcenerschöpfung bedroht sind. So kann ein Angreifer beispielsweise einen Authentisierungsvorgang beginnen und dann den Vorgang beim Server in einem initialen Zustand belassen. Gelingt es einem Angreifer, einen Server dazu zu bringen, eine große Anzahl berechnungsintensiver Operationen – z.B. durch Überprüfung vorgetäuschter Signaturen, Generierung Server-seitiger Signaturen, Diffie-Hellman-Berechnungen etc. – auszuführen, so kann beim Server hierdurch eine Ressourcenerschöpfung eintreten, die dazu führt, dass die Dienste des Servers für legitime Dienstnehmer nicht mehr zur Verfügung stehen (vergleiche Abb. 5).

Diese Überlegungen sollen im folgenden am Beispiel des häufig eingesetzten Netzsicherheitsprotokolls *Transport Layer Security (TLS)* bzw. *Secure Socket Layer (SSL)* verdeutlicht werden (TLS ist die von der IETF genormte Variante des bei Web-Servern und -Browsern gebräuchlichen SSL-Protokolls). Abb. 6 zeigt den Authentisierungsdialog von TLS/SSL im Überblick. In dieser Darstellung wird von den kryptographischen Details abstrahiert, da TLS/SSL eine Vielzahl unterschiedlicher Optionen unterstützt.

Für die Authentisierung und Schlüselaushandlung sieht TLS die folgenden Protokolloptionen vor:

– *Schlüsselübermittlung per RSA:* Bei dieser Methode wird ein sogenanntes Pre-Master-Secret vom Client zufällig erzeugt und mit dem öffentlichen Schlüssel des Servers verschlüsselt an diesen geschickt. Der Server sendet bei dieser Methode kein eigenes KeyExchange-Nachrichtenelement an den Client, da er nicht aktiv an der Erzeugung des gemeinsamen Geheimnisses beteiligt ist.
– *Diffie-Hellman:* Hierbei wird ein herkömmliches Diffie-Hellman-Schlüsselaustauschprotokoll [6] ausgeführt und das Pre-Master-Secret von dem gemeinsamen Geheimnis $g^{x \cdot y}$ mod p abgeleitet.
– *Wiederaufnahme einer Sitzung:* falls der Server die Wiederaufnahme bereits etablierter und unterbrochener Sitzungen unterstützt, so können Client und Server einen verkürzten Dialog führen, bei dem für beide Seiten ein erheblich geringerer Berechnungsaufwand anfällt. In diesem Fall wird aus dem Pre-Master-Secret der unterbrochenen Sitzung neues Schlüsselmaterial abgeleitet.

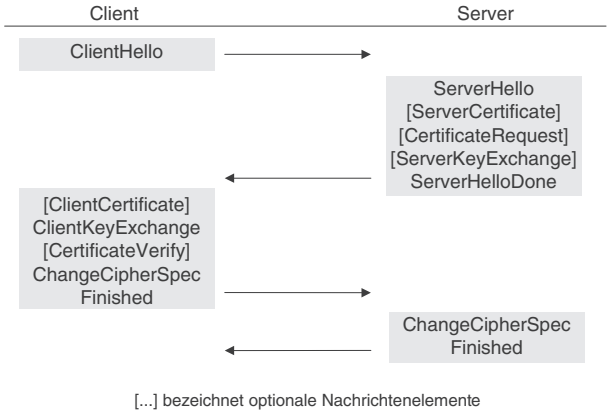


Abb. 6 Überblick über die TLS-Authentisierung

Aufgrund der Tatsache, dass ein DoS-Angreifer in der Regel bestrebt ist, einen möglichst hohen Berechnungsaufwand beim Server hervorzurufen, wird er eher eine der beiden erstgenannten Protokollvarianten auswählen.

Da das weitgehend mit TLS identische SSL ursprünglich für die Sicherung von HTTP-Datenströmen (WWW-Verkehr) entworfen wurde, besteht das übliche Anwendungsszenario darin, dass ein Web-Browser Daten von einem authentischen Web-Server anfordert und daher die Identität des Servers überprüfen möchte. In diesem Fall sendet der Server dem Client nach seiner „Hello"-Nachricht (vergl. Abb. 6) das Zertifikat über seinen öffentlichen Schlüssel. Dieses Zertifikat enthält gegebenenfalls einen öffentlichen RSA-Schlüssel und/oder die öffentlichen Diffie-Hellman-Parameter des Servers. Der Client überprüft daraufhin die Authentizität der Schlüsselparameter des Servers und verwendet sie im Anschluss dafür, mit dem Server RSA- oder Diffie-Hellman-basiert ein Pre-Master-Secret auszuhandeln.

Im Fall einer RSA-basierten Schlüsselaushandlung wählt der Client hierfür zufällig das zu verwendende Pre-Master-Secret aus, verschlüsselt es mit dem öffentlichen Schlüssel des Servers und schickt es an ihn. Der Server dechiffriert darauf hin das Pre-Master-Secret mit seinem privaten Schlüssel, leitet aus dem Pre-Master-Secret zunächst das Master-Secret und daraus die zu verwendenden Sitzungsschlüssel ab und schickt schließlich eine mit dem (neben weiteren Schlüsseln) so konstruierten Integritätsschlüssel signierte Nachricht an den Client. Der Client kann nach Überprüfung dieser Nachricht darauf schließen, dass der Server den korrekten Integritätsschlüssel ableiten konnte und daher in Besitz des zu dem zertifizierten öffentlichen Schlüssel gehörenden privaten Schlüssel sein muss und somit authentisch ist.

Im Fall einer Diffie-Hellman-basierten Schlüsselaushandlung berechnet der Client mit den Diffie-Hellman-Werten des Ser-

vers und seinen eigenen Werten einen entsprechenden Wert und sendet ihn an den Server. Der Server führt die dem Diffie-Hellman-Protokoll gemäßen Berechnungen durch und antwortet mit seinem eigenen Wert. Anschliessend berechnen beide aus den ausgetauschten und eigenen Werten das Pre-Master-Secret. Da die öffentlichen Diffie-Hellman-Werte des Servers zertifiziert sind, kann der Client nach Erhalt einer mit dem abgeleiteten Integritätsschlüssel signierten Nachricht auf die Authentizität des Servers schließen (für eine ausführlichere Darstellung dieses Protokolls siehe [23, Kapitel 12]).

Beide Varianten der Schlüsselaushandlung können von einem Angreifer für die Ausführung von Sabotageangriffen ausgenutzt werden. Im Fall der RSA-basierten Aushandlung sendet der Angreifer eine korrekt formatierte „ClientKeyExchange"-Nachricht, in der vermeintlich ein mit dem öffentlichen Schlüssel des Servers chiffriertes PreMaster-Secret enthalten ist. Tatsächlich führt der Angreiferjedoch keine asymmetrische Verschlüsselung durch, sondern sendet lediglich eine Bitfolge, die wie ein RSA-Schlüsseltext aussieht. Der Server kann erst nach einer aufwändigen RSA-Entschlüsselung feststellen, dass der erhaltene Klartext nicht korrekt formatiert ist, sowie die Nachricht und den bereits angelegten Protokollzustand verwerfen.

In einer ähnlichen Weise kann der Diffie-Hellman-Schlüsselaustausch angegriffen werden. Hierbei sendet der Angreifer irgendeine hinreichend lange Zahl als seinen öffentlichen Diffie-Hellman-Wert. Der Server wird daraufhin seinen Teil der Berechnungen durchführen, der eine auswendige Exponentation mit sehr großen Zahlen einschließt, bevor er an der nicht-korrekten bzw. ausbleibenden „Finished"-Nachricht des Clients erkennen kann, dass der Client nicht korrekt an dem Protokoll teilnimmt. In beiden Fällen gelingt es somit einem DoS-Angreifer, mit relativ geringem Aufwand einen erheblichen Ressourcenkonsum bei dem angegriffenen Server auszulösen.

4.3 Gegenmaßnahmen

Zur Abwehr der im vorigen Abschnitt beschriebenen Angriffstechniken sind bereits eine Reihe von Ansätzen vorgeschlagen worden. Meadows beschreibt in [16] ein formales Rahmenwerk für die Evaluierung DoS-resistenter Protokolle mit dem Ziel den Evaluierungsprozess systematischer zu gestalten. Dabei führt sie auch die Grundidee ein, zu Protokollbeginn zunächst eine „schwache" Authentisierung vorzunehmen und diese dann schrittweise zu erweitern, sofern das erforderlich ist. Damit sollen eine gewisse Rückverfolgbarkeit von DoS-Angriffen erreicht und darüber hinaus potentielle Angreifer demotiviert werden, Angriffe auf der Basis von Ressourcenerschöpfung zu starten, da sie hierfür auch eigene Ressourcen binden müssten. In praktischen Szenarien fand dieser Ansatz bislang allerdings eher geringe Beachtung.

Die meisten in der Praxis favorisierten Abwehrtechniken gegen Sabotageangriffe auf kryptographische Protokolle beruhen auf den folgenden Ideen:

- *Zustandslose Protokollbearbeitung*, bei der die Speicherung von Zustandsinformation beim Server vermieden wird, bis die Authentizität des Clients überprüft werden konnte,
- *Herkunftsprüfung mittels Berechtigungsmarken*, bei denen der Server dem Client zunächst eine Art Berechtigungsmarke (im Englischen in der Regel als Cookie bezeichnet) an die Quelladresse der Anfrage schickt und die Anfrage erst bearbeitet, nachdem der Client diese Marke zurückgeschickt hat, sowie
- *Erhöhung des Angriffsaufwands durch kryptographische Aufgaben*, bei denen der Server dem Client auf eine Anfrage hin zunächst eine für den Client aufwändig zu lösende, für den Server jedoch einfach zu überprüfende Aufgabe (im Englischen „*Client Puzzle*" genannt) stellt, wodurch der Aufwand für einen DoS-Angreifer vom Server lastabhängig in beliebiger Weise erhöht werden kann.

Die genannten Ansätze sollen in den folgenden Abschnitten kurz erklärt werden.

4.3.1 Zustandslose Protokollbearbeitung

Die Grundidee zu Beginn eines Authentisierungsprotokolls auf die Speicherung von Zustand bei den beteiligten Instanzen zu verzichten und die Vorteile dieser Vorgehensweise wurde von Yung et. al. in [13] beschrieben. Aura und Nikander verallgemeinerten diesen Ansatz hin zu zustandslosen Servern, die ihre Verbindungen durch Verlagerung des gesamten Zustands hin zu den Clients verwalten [1].

Das generelle Ziel dieses Ansatzes besteht darin zu vermeiden, dass der Server Zustandsinformation speichern muss, solange nicht entschieden werden kann, ob sich hinter einer Anfrage ein Sabotageangriff verbirgt. Aura und Nikander schlagen daher vor, dass die ersten Schritte eines Authentisierungsprotokolls vom Server zustandslos bearbeitet werden sollen, bis er die Authentizität des Clients überprüft hat bzw. sich durch andere Maßnahmen von der Aufrichtigkeit der Client-Anfrage überzeugt hat (siehe hierzu auch den unten stehenden Abschnitt zur Erhöhung des Angriffsaufwands).

Diese Grundidee wird in Abb. 7 illustriert: Anstatt den für einen Protokollablauf erforderlichen Zustand beim Server zu speichern, wird er nach jedem Schritt hin zum Client übertragen und von diesem mit der nächsten Nachricht wieder zurückgeschickt. Um die Zustandsdaten vor unberechtigter Einsichtnahme und Modifikation zu schützen, sollten angemessene Sicherheitsmaßnahmen (Verschlüsselung und Integritätssicherung) getroffen werden. Da lediglich der Server die Zustandsinformation lesen und bearbeiten soll, kann hierfür auf symmetrische Kryptographie zurückgegriffen werden, wodurch eine effiziente Implementierung der Sicherung auch ohne aufwändiges Schlüsselmanagement möglich ist.

Zustandsbehaftete Bearbeitung		Zustandslose Bearbeitung
1. C → S: Msg_1	S speichert $State_{s1}$	1. C → S: Msg_1
2. S → C: Msg_2		2. S → C: Msg_2, $State_{s1}$
3. C → S: Msg_3	S speichert $State_{s2}$	3. C → S: Msg_3, $State_{s1}$
4. S → C: Msg_4		4. S → C: Msg_4, $State_{s2}$

Abb. 7 Prinzip der zustandslosen Protokollbearbeitung

Zustandslose Protokolle erweisen sich als erheblich resistenter gegen Angriffe, die darauf abzielen, bei einem Server Authentisierungs- oder Verbindungsaufbauvorgänge in einem „halb-offenen" Zustand zu belassen, da bei ihnen der Zustand nicht im Server, sondern im Netzwerk gespeichert" wird. Für diesen Vorteil müssen jedoch die Nachteile eines erhöhten Bandbreitebedarfs und eines möglichen Verlusts von Zustandsdaten im Netzwerk in Kauf genommen werden.

4.3.2 Herkunftsprüfung

Die Grundidee der Herkunftsprüfung mittels anfragespezifischer Berechtigungsmarken wurde erstmals von Karn und Simpson im sogenannten *Photuris*-Protokoll [19, 20] auf Authentisierungsprotokolle angewendet und fand im Anschluss Eingang in das Authentisierungsprotokoll *ISAKMP* [15, 10], der *IPSec*-Sicherheitsarchitektur für das Internet Protokoll. Eine solche Berechtigungsmarke („*Cookie*“) ist eine vom Server generierte Datenstruktur, die zu Beginn eines Protokollablaufs zur Absenderadresse der Anfrage geschickt wird und vom Client in die weiteren Nachrichten des Protokollablaufs jeweils integriert werden muss. Auf diese Weise kann der Server sichergehen, dass sich sein Kommunikationspartner zumindest auf der Route zwischen Server und vorgegebener Initiatoradresse befinden muss, da er andernfalls nicht in Besitz der anfragespezifischen Berechtigungsmarke gelangen könnte.

In der Spezifikation des Photuris-Protokolls [19] werden die folgenden Anforderungen an die Herkunftsprüfung gestellt:

- die Berechtigungsmarke muss von der Quell- und der Zieladresse der Anfrage abhängen,
- es darf einem Angreifer nicht möglich sein, eine Berechtigungsmarke zu fälschen, die vom Server akzeptiert würde,
- die Erzeugung und Überprüfung von Berechtigungsmarken muss so effizient realisierbar sein, dass sie kein neues Realisierungspotential für Sabotageangriffe eröffnet, und
- der Server sollte keinen Client-spezifischen Zustand speichern, bevor er nicht eine entsprechende, von ihm selbst generierte Berechtigungsmarke vom Client zurückgeschickt bekommen hat.

Für die Erzeugung entsprechender Berechtigungsmarken empfiehlt die Photuris-Spezifikation die Verwendung der kryptographischen Hash-Funktion MD5. Mit dieser Funktion soll ein Hash-Wert über ein nur dem Server bekanntes Geheimnis, die IP-Quell- und -Zieladresse sowie den Quell- und den Zielport generiert werden.

Zusammenfassend erlaubt die Herkunftsprüfung mittels anfragespezifischer Berechtigungsmarken eine effiziente und effektive Abwehr von Angriffen auf Authentisierungsprotokolle mit gefälschten Quelladressen, da der Angreifer in diesem Fall die vom Server gesendete Berechtigungsmarke mit hoher Wahrscheinlichkeit nicht erhalten wird und somit keinen erheblichen Ressourcenkonsum beim Server hervorrufen kann. Befindet sich der Angreifer jedoch auf dem Pfad zwischen dem Server und der vorgetäuschten Quelladresse, so kann er die Berechtigungsmarke abfangen und der Schutzmechanismus versagt.

Weiterhin kann die Herkunftsprüfung auf effiziente Weise mit zustandsloser Protokollbearbeitung verbunden werden, da bei zustandsloser Protokollverarbeitung ohnehin der integritätsgeschütze Zustand vom Server zum Client und zurück übertragen wird.

4.3.3 Erhöhung des Angriffsaufwands

Die Idee, die Senderate eines potentiellen Angreifers durch Stellen kryptographischer Aufgaben – sogenannter „*Client Puzzles*“ – zu drosseln, wurde erstmals von Dwork und Naor vorgeschlagen, die mit dieser Maßnahme sogenannten *Spam-Emails* vorbeugen wollten [7]. Juels und Brainard präsentierten in [14] eine einfachere Aufgabe, die dem Initiator einer TCP-Verbindung während der Verbindungsaufbauphase aufgegeben werden kann, um TCP/SYN-Angriffe zu bekämpfen. Aura et. al. erweiterten diese Idee für den Schutz kryptographischer Protokolle [2]. Die folgende Darstellung folgt weitgehend der letztgenannten Referenz.

Die grundlegenden Anforderungen an kryptographische Aufgaben zur DoS-Bekämpfung wurden von Aura et. al. wie folgt formuliert:

- die Erzeugung und Überprüfung der Aufgabe erfordert vom Server nur geringen Berechnungsaufwand,
- der Server kann den vom Client zu erbringenden Aufwand auf einfache und annähernd kontinuierliche Weise zwischen „kein Aufwand“ und „unmöglich zu lösen“ einstellen,
- die Aufgabe kann auf unterschiedlichen Hardwareplattformen gelöst werden,
- die Vorausberechnung von Lösungen ist für den Client unmöglich,
- der Server muss keine Daten speichern während der Client die Aufgabe löst,
- die gleiche Aufgabe kann gleichzeitig an mehrere Clients ausgegeben werden, wobei die Kenntnis einer oder mehrerer Lösungen einem neuen Client bei der Lösung der Aufgabe nicht hilft, und
- ein Client kann eine Aufgabe wiederverwenden, indem er mehrere Instanzen der Aufgabe erzeugt; hierbei ist es ihm jedoch nicht möglich, die Lösung einer neuen Instanz aus bereits vorhandenen Lösungen der Aufgabe zu berechnen.

Aura et. al. beschreiben in [3] das folgende Schema, das die genannten Anforderungen erfüllt und auf der teilweisen Umkehr einer kryptographischen Hashfunktionen per erschöpfender Suche beruht: Der Server wählt periodisch eine hinreichend große Zufallszahl N_S (z.B. 128 bit lang) und sendet diese zusammen mit dem Schwierigkeitsgrad k der Aufgabe per Broadcast oder auch als Reaktion auf konkrete Anfragen an interessierte Clients. Ein Client C der eine solche Aufgabe lösen will, um damit die Bearbeitung seiner Anfrage zu erlangen, wählt eine eigene Zufallszahl N_C und ermittelt dann per erschöpfender Suche einen Wert X, so dass gilt:

$$h(C, N_S, N_C, X) = \underbrace{000\ldots000}_{k} Y$$

Der Wert Y repräsentiert dabei einen beliebigen Wert für die restlichen Bits des kryptographischen Hash-Werts. Der Client sendet nach Berechnung von X das Tupel (C, N_S, N_C, X) an den Server. Der Server überprüft daraufhin, ob die ersten k Bits des Hashwerts $h(C, N_S, N_C, X)$ tatsächlich den Wert „0“ aufweisen und ob der Client C diese Lösung bereits einmal präsentiert hat (im letzteren Fall würde die Lösung zurückgewiesen). Verlaufen beide Prüfungen erfolgreich, so speichert der Server die Lösung bis zur Ausgabe einer neuen Zufallszahl N_S und fährt mit der weiteren Protokollbearbeitung fort. Bei steigender Last kann der Server den Schwierigkeitsgrad der Aufgabe durch Erhöhen des Werts k relativ feingranular anpassen, da sich mit Inkrementierung von k um 1 der vom Client zu erbringende Rechenaufwand jeweils verdoppelt. Ebenso kann der Server bei niedriger Last die Clients durch Verringern von k entlasten.

Abb. 8 zeigt die Integration dieses Schemas in ein Authentisierungsprotokoll mit aufwändiger Signaturprüfung (z.B. per RSA- oder ElGamal-Algorithmus). Hierbei überprüft der Server zunächst die vom Client präsentierte Lösung der kryptographi-

schen Aufgabe bevor er erhebliche Ressourcen für die Signaturprüfung aufwendet.

Dean und Stubblefield beschreiben die Integration eines ähnlichen Schemas in das TLS/SSL-Protokoll [5]. Als Reaktion auf die ClientHello-Nachricht sendet der Server zunächst die Parameter seiner Aufgabe an den Client und fährt erst nach Erhalt einer neuen und korrekten Lösung mit der weiteren (berechnungsaufwändigen) Bearbeitung des TLS/SSL-Protokolls fort.

Zusammenfassend stellen kryptographische Aufgaben (Client Puzzles) ein effektives Mittel zur Reduzierung der Angriffsrate einzelner DoS-Angreifer dar. Die grundlegende Idee hierbei ist, dass ein Client zunächst eigene Ressourcen für eine Anfrage einbringen muss, bevor der Server seine Ressourcen für die Bearbeitung der Anfrage aufwendet.

Client		Server
		S wählt periodisch k, generiert N_S, Zeitstempel T_S und signiert die Nachricht
	$S_S(T_S,k,N_S)$ ⟵	
C verifiziert den Zeitstempel T_S und Signatur S_S. C generiert N_C und bestimmt eine Lösung X, so dass: $h(C,N_S,N_C,X) = O_1O_2…O_KY$. C signiert die Nachricht		
	$S_C(S,C,N_S,N_C,X)$ ⟶	S verifiziert, dass N_S aktuell ist, C, N_S, N_C noch nicht benutzt wurden, und dass $h(C,N_S,N_C,X)=O_1,O_2…O_KY$. **S kann nun Ressourcen für C aufbringen**. S speichert C,N_S,N_C solange N_S aktuell ist und verifiziert die Signatur S_C. **S hat nun C authentisiert**. S signiert die Nachricht
	$S_S(S,C,N_C)$ ⟵	
C verifiziert die Signatur S_S. C hat nun S authentisiert.		

Abb. 8 Protokollintegration eines Client Puzzle

5 ZUSAMMENFASSUNG

Aus dem hohen Risikopotential von Sabotageangriffen für jetzige wie zukünftige Netzinfrastrukturen und der wachsenden Abhängigkeit unserer modernen Informationsgesellschaft von der Verfügbarkeit dieser Netze, ergibt sich eine ständig steigende Bedrohung, der angemessen entgegnet werden muss. Das gilt in verstärktem Maße bei Vereinheitlichung der verwendeten Kommunikationsprotokolle, wie sie etwa derzeit durch die zunehmende Einführung IP-basierter Komponenten in die Netzinfrastruktur angestrebt wird (siehe z.B. zukünftige Releases der UMTS-Standards). Es besteht somit ein dringender Bedarf dafür, systematische Bedrohungsanalysen durchzuführen und einen abgestimmten Maßnahmenkatalog zu entwickeln, der es erlaubt, DoS- und DDoS-Angriffen effektiv zu begegnen und der sowohl kryptographische Maßnahmen (Cookies, Client Puzzles) als auch netzwerktechnische Maßnahmen (Traceback, Paketfilterung, Intrusion Detection, aktive Netztechnologien etc.) umfassen wird.

LITERATUR

[1] Aura, T.; Nikander, P.; Leiwo, J.: Stateless Connections. In: Proceedings of the International Conference an Information and Communications Security (ICIS). Springer, 1997. Lecture Notes in Computer Science (LNCS).

[2] Aura, T.; Nikander, P.; Leiwo, J.: Towards Network Denial of Service Resistant Protocol. In: Proceedings of the 15th International Information Security Conference (IFIPISEC '2000). Kluwer, Aug. 2000. Beijing, China.

[3] Aura, T.; Nikander, P.; Leiwo, J.: DOS-ResistantAuthentication with Client Puzzles. In: Proceedings of the Security Protocols Workshop 2000. Springer, 2001. Cambridge, UK, April 2000, Lecture Notes in Computer Science (LNCS).

[4] CERT: W32/Blaster Worm. http://www.cert.org/advisories/CA-2003-20.html, August 2003.

[5] Dean, D.; Stubblefield, A.: Using Client Puzzles to Protect TL S. In: Proceedings of the 10th Annual USENIX Security Symposium, 2001.

[6] Diffie, W.; Hellman, M.E.: New Directions in Cryptography. IEEE Transactions an Information Theory, S. 644-654, 1976.

[7] Dwork, C.; Naor, M.: Pricing via Processing-or-Combatting Junk Mail, 1993. Lecture Notes in Computer Science (LNCS) No. 740.

[8] •••• (ed.): Results of the Distributed-Systems Intruder Tools Workshop. White Paper, CERT Coordination Center, Dez. 1999.

[9] Elgamal, T.: A Public Key Cryptosystem and a Signature Scheme based an Discrete Logarithms. IEEE Transactions on Information Theory, 31 (4): 469-472, Juli 1985.

[10] Harkins, D.; Carrel, D.: The Internet Key Exchange (IKE), Nov. 1998. RFC 2409, IETF, Status: Proposed Standard, ftp://ftp.internic.net/rfc/rfc2409.txt.

[11] Howard, J.D.: An Analysis of Security Incidents on the Internet. PhD thesis, Carnegie Mellon University, Aug. 1998.

[12] Institute, C.S.; F. B. of Investigation: 1999 CSI/FBI Computer Crime and Security Survey. Computer Security Institute Publication, März 1999.

[13] Janson, P.; Tsudik, G.; Yung, M.: Scalability and Flexibility in Authentication Services: The KryptoKnight Approach. In: Proceedings of IEEE INFOCOM '97, Apr. 1997. Tokyo, Japan.

[14] Juels, A.; Brainard, J.: Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks. In: Proceedings of the 1999 Network and Distributed System Security Symposium (NDSS '99), März 1999. Internet Society.

[15] Maughan, D.; Schertler, M.; Schneider, M.; Turner, J.: Internet Security Association and Key Management Protocol (ISAKMP), Nov. 1998. RFC 2408, IETF, Status: Proposed Standard, ftp://ftp.internic.net/rfc/rfc2408.txt.

[16] Meadows, C.: A Formal Fiamework and Evaluation Method for Network Denial of Service. In: PCSFW. Proceedings of The 12th Computer Security Foundations Workshop. IEEE Computer Society Press, 1999.

[17] Menezes, A.; Van Oorschot, P.; Vanstone, S.: Handbook of Applied Cryptography. CRC Press LLC, 1997.

[18] Moore, D.; Voelker, G.M.; Savage, S.: Inferring Internet Denial-of-Service Activity. In: Usenix Security Symposium, S. 9-22, 2001.

[19] Karn, P.; Simpsonk, W.: Photuris: Session-Key Management Protocol. RFC 2522 (Informational), März 1998.

[20] Karn, P.; Simpsonk, W.: Photuris: Extended Schemes and Attributes. RFC 2523 (Informational), März 1999.

[21] Rivest, R.; Shamir, A.; Adleman, L.: A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Communications of the ACM, Feb. 1978.

[22] Savage, S.; Wetherall, D.; Carlin, A.; Anderson, T.: Practical Network Support for IP Traceback. In: ACM SIGCOMM, S. 295-306, Okt. 2000.

[23] Schäfer, G.: Netzsicherheit – Algorithmische Grundlagen und Protokolle. dpunkt.verlag, 2003.

[24] Song, D.X.; Perrig, A.: Advanced and Authenticated Marking Schemes for IP Traceback. In: Proceedings IEEE Infocomm 2001, 2001.

[25] Staniford, S.; Grim, G.; Jonkman, R.: Flash Worms: Thirty Seconds to Infect the Internet. http://www. silicondefense.com/flash/.

[26] Team, C.E.R.: CERT Advisory CA-2000-01: Denial-of-Service Developments. http://www.cert.org/advisories/CA-2000-01.html, Jan. 2000.

[27] Team, C.E.R.: CERTICC Statistics 1988-2003. http://www.cert.org/stats, 2003.

[28] Weaver, N.C.: Warhol Worms: The Potential for Very Fast Internet Plagues. http://www.cs.berkeley.edu/-nweaver/warhol.html.